Veres Péter[1] – Bányai Tamás[2] – Illés Béla[3]

HÁLÓZATSZERŰ SZOLGÁLTATÓ RENDSZER TERVEZÉSE[4]

A globalizáció hatásai nem csupán a termelés, hanem a szolgáltatás területén olyan változásokat idéztek elő, melyek szükségessé tették a szolgáltatási tevékenységek hálózatosodását. Ez ahhoz vezetett, hogy napjaink szolgáltatási rendszereiben egyre összetettebb folyamatok jelennek meg és az ezeket kiszolgáló, támogató logisztikai tevékenységek is egyre komplexebbekké válnak. Ezen komplex logisztikai folyamatok tervezése olyan újszerű, bonyolult és gyakran NP-hard[5] problémák megoldását támogató modellek, módszerek és algoritmusok alkalmazását követeli meg, melyek nagyméretű állapotterekben is képesek elfogadható megoldást előállítani a felvetődő tervezési feladatok megoldásához. Jelen cikkben a szerzők egy olyan firefly algoritmuson[6] alapuló heurisztikus optimálási módszert mutatnak be, mely alkalmas a hálózatszerűen működő logisztikai folyamatok optimális kialakításának támogatására. Bemutatásra kerül egy általános modell, illetve az annak megoldására szolgáló algoritmus, melyben a szerzők olyan új metrikát vezetnek be a permutációk közötti távolságok mérésére.

DESIGN OF NETWORKED SERVICES

The globalisation of economy and market leaded to increased networking in the field of manufacturing and services. The processes of these manufacturing and services including logistics became more and more complex. The design and operation of these complex processes can be described as NP-hard optimisation problems. These problems can be solved using sophisticated modelling, methods metaheuristics based algorithms. Much of the research in this area is focusing on manufacturing. This paper aims to report a firefly metaheuristics based optimisation method, by the aid of which it is possible to support the solution of design and control problems of networked service processes. The authors describe a general model and present a new metrics to measure permutation distances used in the algorithm.

BEVEZETÉS

A logisztika a termelési folyamatok mellett egyre nagyobb szerepet tölt be a szolgáltatási tevékenységek területén is. A termelési folyamatok tervezésére rengetek szakirodalom áll rendelkezésre, azonban a szolgáltatási tevékenység kapcsán még jelentős területek vannak, melyek esetében az optimális kialakítás és működtetés támogatására nem állnak rendelkezésre megfelelő modellek, módszerek és algoritmusok [1]. Ezen területen jelentős eredményeket felmutató kutatói team a Lost in Services kutatói team, mely műszaki és matematikai modelleket dolgozott ki szolgáltatási folyamatok tervezésére és működtetésére [2]. Ebben a kutatási irányban olyan perspektívák mutatkoznak, melyek komoly haszonnal kecsegtetnek a szolgáltatási folyamatok kialakítása és működtetése során történő alkalmazás esetében.

[1] doktorandusz, Miskolci Egyetem, altveres@uni-miskolc.hu
[2] egyetemi docens, Miskolci Egyetem, alttamas@uni-miskolc.hu
[3] intézetigazgató egyetemi tanár, Miskolci Egyetem, altilles@uni-miskolc.hu
[4] Lektorálta: Prof. Dr. Mang Béla, egyetemi tanár, Miskolci Egyetem, bela.mang@uni-miskolc.hu
[5] Nem polinomiális nehéz.
[6] A szentjánosbogarak szociális viselkedésén alapuló heurisztikus optimálási módszer.

Jelen munka célja egy olyan általános, metaheurisztikán alapuló tervezési módszer bemutatása, mely alkalmas a termelési folyamatokban szerzett tapasztalatok hasznosítása révén a szolgáltatási rendszerek logisztikai folyamatainak optimális kialakítását támogatni.

IRODALMI ÁTTEKINTÉS

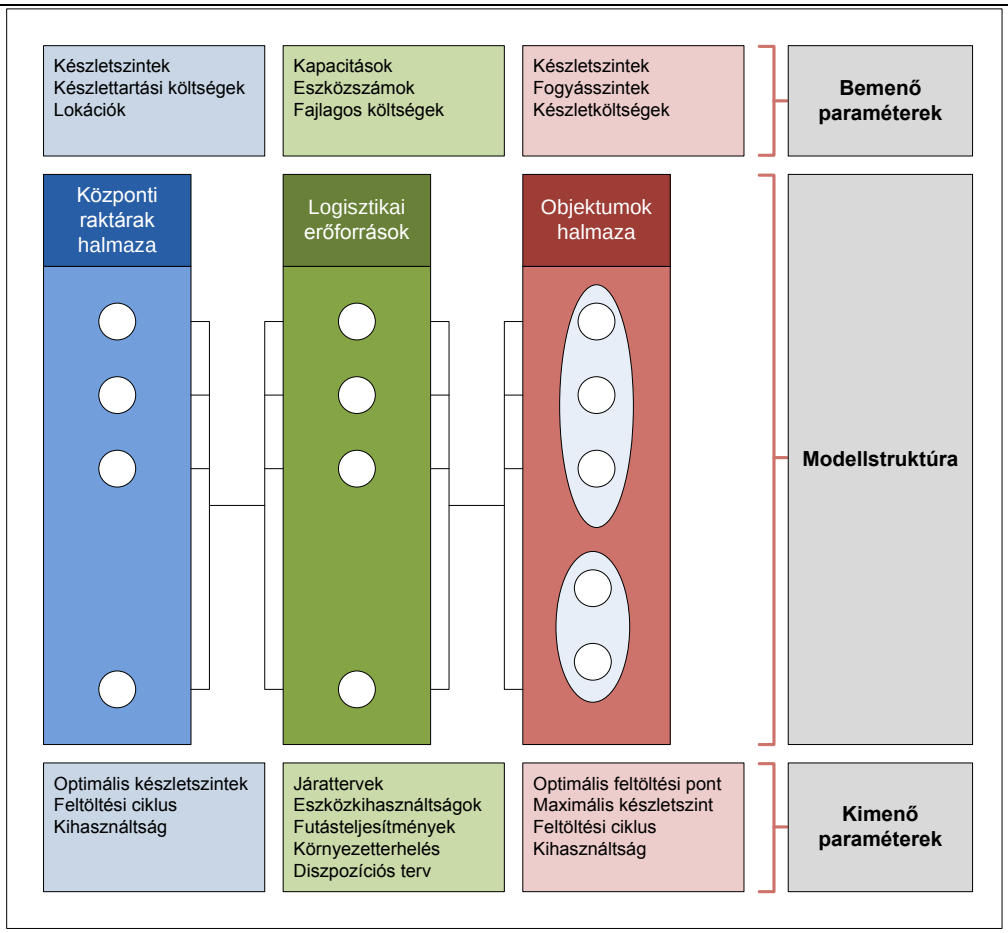
A szakirodalomban számos olyan kutatási munka található, mely a szolgáltatási tevékenységekhez kapcsolódó logisztikai folyamatok tervezésének és irányításának kérdéseit tárgyalja. Ezen irodalmak egy része csupán koncepció szinten vizsgálja a hálózatszerűen működő logisztikai folyamatok optimális kialakítását [3], míg másik részük konkrét modellekkel és algoritmusokkal szolgál a tervezési feladatok megoldásához [4][5][6]. A szolgáltatási területek egyik logisztikai tevékenységekben leggazdagabb területe a city logisztika, mely az áruelosztás problémakörében szociális, kulturális és gazdasági hatásokkal bírhat, így tervezése különösen nagy jelentőséggel bír [7][8]. A logisztikai szolgáltatások optimális kialakítása nem csupán a hagyományos logisztikai funkcionális területeket érinti (beszerzés, termelés, elosztás, újrahasznosítás). Jelentős kutatások folynak azzal a céllal, hogy olyan módszereket és algoritmusok dolgozzanak ki, melyekkel a logisztikai szolgáltatások kialakítását úgy lehet támogatni, hogy a teljes logisztikai szolgáltató lánc környezetterhelése jelentős mértékben csökkenthető [9]. A hálózatszerűen működő szolgáltatási tevékenység vizsgálatának egy érdekes aspektusa a minőségbiztosítás és a kockázatelemzés kérdése, ugyanis a legtöbb szakirodalmi forrás a hálózatszerűen működő rendszerek esetében is csak az egyes résztvevők kockázatát vizsgálja, kevés olyan kutatási munka található, mely egy logisztikai szolgáltató hálózat kockázatát rendszerszinten kezeli [10]. A logisztikai tevékenységek szervezésekor a környezetvédelmi hatások figyelembevétele elkerülhetetlen, s ez különösen igaz az olyan szolgáltatási tevékenységeket támogató logisztikai folyamatok esetében, ahol nagy fajlagos szállítási költségek és teljesítmények adódnak a folyamatok jellegéből adódóan [11]. A szakirodalmi háttér egyértelműen indokolja egy olyan modell és módszer kidolgozását, melynek segítségével a hálózatszerűen működő szolgáltatási folyamatokat kiszolgáló logisztikai rendszerek optimalizálhatóak.

A MODELL

Jelen kutatómunka célja egy olyan modell megalkotása, mely alkalmas hálózatszerűen működő szolgáltatási rendszerek működtetéséhez kapcsolódó logisztikai folyamatok leírására. A modellezés során a célunk egy olyan általános modell megfogalmazása volt, melyből különböző korlátozások figyelembe vételével speciális ágazati modellek is megalkothatóak. A modellben három olyan elemcsoportot szerepeltetünk, melyek alapján a szolgáltatási tevékenységet támogató logisztikai folyamat modellezhető:

- központi raktárak,
- logisztikai erőforrások,
- kiszolgálandó objektumok.

Az 1. ábrán bemutatott modellből látható, hogy többek között az egyes elemcsoportok közötti kapcsolatok meghatározása fogja az optimalizálás tárgyát képezni.



1. ábra A hálózatszerűen működő szolgáltatási rendszer egy általános modellje

Az 1. ábrán vázolt modellben először fogalmazzuk meg a célfüggvényeket. Az első célfüggvény az egyes járatok által megtett úthossz minimalizálása. Ezen célfüggvényben a központi raktár és a körjárat első objektuma, az egyes objektumok közötti, illetve az utolsó objektum és a központi raktár közötti úthosszak járatokra és vizsgálati időintervallumokra vonatkozó értékét kell összegezni.

$$L = \sum_{t=1}^{\tau} \sum_{\alpha=1}^{\delta(t)} \left(l^{KR}_{f(t,\alpha),p_{t,\alpha(t),1}} + \sum_{s=1}^{m(t,\alpha)-1} l_{p_{t,\alpha(t),s},p_{t,\alpha(t),s+1}} + l^{KR}_{p_{t,\alpha(t),m,f(t,\alpha)}} \right) \rightarrow min. \quad (1)$$

ahol τ a vizsgálati időszak ciklusainak száma,

α járatazonosító,

$\delta(t)$ járatok száma a vizsgálati időszak t-edik vizsgálati ciklusában,

$f(t,\alpha)$ a vizsgálati időszak t-edik vizsgálati ciklusában az α-adik járathoz rendelt központi raktár azonosítója,

β objektum sorrend azonosító egy adott járatban,

$p_{t,\alpha,\beta}$ a vizsgálati időszak t-edik vizsgálati ciklusában az α-adik járathoz rendelt központi raktárból β-ként felkeresett objektum azonosítója,

$m(t,\alpha)$ a vizsgálati időszak t-edik vizsgálati ciklusában az α-adik járat által felkeresett objektumok száma.

Az úthosszra vonatkozó célfüggvényből a minimális anyagmozgatási költségre vonatkozó célfüggvény egyértelműen meghatározható:

$$C = L \cdot c^L \rightarrow min. \tag{2}$$

ahol c^L a járat fajlagos, úthossztól függő költsége.

Az egyes járatok kihasználtsága a következő célfüggvény, amit a rendszer optimális kialakítása során figyelembe kell venni. Ezen célfüggvény esetében a teljes vizsgálati időtartamra összegezzük az egyes járatok kihasználtságát az egyes objektumokba történő kiszállítási mennyiségek függvényében.

$$\eta^J = \sum_{t=1}^{\tau} \sum_{\alpha=1}^{\delta(t)} \frac{\sum_{\beta=1}^{m(t,\alpha)} K_{p_{t,\alpha,\beta}}^{max} - K_{p_{t,\alpha,\beta}}^{akt}}{K_\alpha^J} \rightarrow max. \tag{3}$$

ahol $K_{p_{t,\alpha,\beta}}^{max}$ a vizsgálati időszak t-edik vizsgálati ciklusában az α-adik járathoz rendelt központi raktárból β-ként felkeresett objektum maximális tárolókapacitása,

$K_{p_{t,\alpha,\beta}}^{akt}$ a vizsgálati időszak t-edik vizsgálati ciklusában az α-adik járathoz rendelt központi raktárból β-ként felkeresett objektum aktuális tárolókapacitása,

K_α^J az α-adik járathoz rendelt jármű kapacitása.

Fontos célfüggvény a beruházási költségek szempontjából a járatszámok minimalizálása.

$$\alpha_{max} \rightarrow min. \tag{4}$$

A központi raktár fontos szerepet tölt be a rendszerben, ezért az optimalizálás során szükséges figyelembe venni az ellátási láncban betöltött szerepe kapcsán felmerülő szállítási és tárolási költségeket és ezek összegét kell minimalizálni. Meghatározhatjuk a szállítási költséget, mely jelen esetben mint megrendelés feldolgozási költség kerül meghatározásra.

$${}^{KR}K^{MF} = \frac{\tau}{T^C} \cdot c^{SZ} \tag{5}$$

ahol T^C a központi raktárak feltöltési ciklusideje,

c^{SZ} központi raktárak feltöltéséhez kapcsolódó fajlagos megrendelés-feldolgozási költség.

A tárolási költség a teljes vizsgálati időhorizontra szintén számítható.

$${}^{KR}K^T = \sum_{t=1}^{T^C} \sum_{\alpha=1}^{\delta(t)} \sum_{\beta=1}^{m(t,\alpha)} K_{p_{t,\alpha,\beta}}^{max} - K_{p_{t,\alpha,\beta}}^{akt} \tag{6}$$

Ezen két költség ismeretében a központi raktárra vonatkozó célfüggvény a következő alakban írható fel.

$${}^{KR}K = {}^{KR}K^{MF} + {}^{KR}K^T \rightarrow min. \tag{7}$$

A célfüggvények mellett különböző korlátozásokat szükséges figyelembe venni. Az első korlátozás azt fogalmazza meg, hogy az egyes járatok esetében előírható egy minimális és egy maximális úthossz.

$$L_\alpha^J \leq L_{max}^J \quad \text{és} \quad L_\alpha^J \geq L_{min}^J \tag{8}$$

Fontos feltétel, hogy egy adott vizsgálati időpontban egy objektum csak egy központi járatból és egy szállító járművel szolgálható ki, azaz a P mátrixnak permutáció mátrixnak kell lennie egy rögzített t időpontban.

A járatokra is felírható egy fontos korlátozás, mely azt fejezi ki, hogy a járatok által kiszolgált objektumokba kiszállítandó áru mennyisége nem lépheti túl a szállító jármű kapacitását.

$$\sum_{\beta=1}^{m(t,\alpha)} K_{p_{t,\alpha,\beta}}^{max} - K_{p_{t,\alpha,\beta}}^{akt} \leq K_{\alpha}^{J} \tag{9}$$

TÁVOLSÁGMÉRÉS KERESÉSI TÉRBEN

A kidolgozásra kerülő algoritmusban vizsgáljuk az egyes megoldási változatokat reprezentáló permutációk közötti távolságokat definiáló metrikákat, hiszen azok befolyásolhatják az algoritmus konvergenciáját. A heurisztikus és metaheurisztikus optimumkereső algoritmusok esetében különböző állapotterekben kell a lehetséges megoldásváltozatok közül a legjobbat megkeresni. A raj intelligencia típusú algoritmusok esetében (hangya kolónia algoritmus, méh kolónia algoritmus, szentjános bogár algoritmus) az állapottér függvényében különböző módszerek használhatóak az egyes megoldások közötti távolságok mérésére [12]. Amennyiben a keresési térben az egyes megoldási változatok bináris kódoltak, akkor a Hamming távolság egy alkalmas metrika. Valós vektorokkal leírt megoldási változatok esetében Euklidészi vagy Csebisev távolság használható távolságmérésre. Jelen kutatási munkában diszkrét számokat tartalmazó vektorok írják le az egyes megoldási változatokat, így a továbbiakban áttekintjük az ismert metrikákat, majd javaslatot teszünk két új metrika alkalmazására.

Permutációk távolságának mérése

Különböző módszereket tárgyal a szakirodalom a permutációk közötti távolságok mérésére. Az egyik legegyszerűbb módszer a Hamming távolság [13], mely a nem azonos elemeket tartalmazó pozíciók száma két permutációban:

$$d_{Ham}(s_1, s_2) = \sum_{i=1}^{n} x_i \quad ahol \;\; x_i = \begin{cases} 0 \; ha \; s_1(i) = s_2(i) \\ 1 \; más \; esetekben \end{cases} \tag{10}$$

A Hamming távolság normalizálható, ebben az esetben a két permutáció valós Hamming távolságát osztani kell a maximális távolsággal:

$$d_{Ham}^{*}(s_1, s_2) = \frac{d_{Ham}(s_1, s_2)}{n} \tag{11}$$

A különbség távolság a két permutáció egyes elempárjai közötti távolságok különbségeinek összege [14]:

$$d_{különbség}(s_1, s_2) = \sum_{z=1}^{n} |s_1(z) - s_2(z)| \tag{12}$$

A különbség távolság normalizált értéke az egyes permutáció vektorok elemszámának függvényében az

$$d_{különbség}^{*}(s_1, s_2) = \frac{2 \cdot d_{különbség}(s_1, s_2)}{n^2} \; és \; d_{különbség}^{*}(s_1, s_2) = \frac{2 \cdot d_{különbség}(s_1, s_2)}{n^2 - 1} \tag{13}$$

összefüggésekkel számítható.

Amennyiben az egyes permutációk közötti távolságokat nagy távolságok esetében fokozottan kell figyelembe venni, akkor a négyzetes távolságok alkalmazása lehet célszerű:

$$d_{négyzetes különbség}(s_1, s_2) = \sum_{z=1}^{n}\left(s_1(z) - s_2(z)\right)^2 \quad (14)$$

melynek normalizált értéke a

$$d^*_{négyzetes különbség}(s_1, s_2) = \frac{3 \cdot d_{különbség}(s_1, s_2)}{n^3 - n} \quad (15)$$

összefüggéssel számítható.
TSP[7] esetében a leghosszabb közös string, mint távolság igen jól használható, hiszen segítségével azt mérhetem, hogy milyen hosszú azon rész körútnak a hossza, mely megegyezik a két permutáció esetében. Mivel jelen tudományos munkába TSP megoldása a cél, ezért kidolgozásra került két olyan új metrika, mely kifejezetten a TSP típusú feladatok esetében alkalmas a permutációk közötti távolságok leírására.

Új módszerek permutációk távolságának mérése

Az utazó ügynök típusú feladatok esteében igen gyakran jelentkezik igényként az, hogy bizonyos pozíciókat vagy felkeresendő objektumokat prioritással vegyünk figyelembe. Amennyiben a felkeresési sorrendek esetében szükséges prioritásokat alkalmazni, akkor a súlyozott különbség metrika jó alkalmazható:

$$d_{súlyozott\ különbség}(s_1, s_2) = \sum_{z=1}^{n}|s_1(z) - s_2(z)| \cdot w_z \quad ahol \quad \sum_{z=1}^{n} w_z = 1 \quad (16)$$

Ezen módszert lehet négyzetes különbség esetében is alkalmazni:

$$d_{súlyozott négyzetes}(s_1, s_2) = \sum_{z=1}^{n}\left(s_1(z) - s_2(z)\right)^2 \cdot w_z \quad ahol \quad \sum_{z=1}^{n} w_z = 1 \quad (17)$$

A másik általunk megalkotott metrika esetében definiálni kell egy határeltérést, és ezt követően összegezni kell a határeltérésnél nagyobb eltérések számát:

$$d_{határeltérés}(s_1, s_2) = \sum_{i=1}^{n} x_i \quad ahol \quad x_i = \begin{cases} 1\ if\ |s_1(i) - s_2(i)| > dev_{határ} \\ 0\ egyébként \end{cases} \quad (18)$$

AZ ALGORITMUS

Jelen fejezetben a fentiekben vázolt optimalizálási probléma megoldására kidolgozott firefly metaheurisztikán alapuló optimalizálási algoritmus kerül bemutatásra. A Firefly algoritmus egy természet által inspirált metaheurisztikus algoritmus, amelyet a szentjánosbogarak viselkedésének megfigyelése alapján alkottak meg. A szentjánosbogarak fényfelvillanásokon alapuló jelzésrendszert használnak éjszaka figyelemfelkeltetés céljából, amellyel magukhoz vonzzák az ellentétes nemű szentjánosbogarakat. Xin-She Yang alkotta meg az algoritmus első változatát 2008-ban, a következőképpen fogalmazta meg az algoritmus alaptételeit: (1) Minden szentjánosbogár uniszex, tehát mindegyik vonzó hatással van az összes többi egyedre; (2) A vonzás mértéke egyenesen arányos a fényességükkel. Bármelyik két szentjánosbogárra igaz hogy a

[7] Utazó ügynök probléma.

fényesebb vonzza magához a kevésbé fényeset. Viszont a bogarak egymáshoz viszonyított fényessége a távolságukkal arányos; (3) A legfényesebb bogár véletlenszerűen mozdul el, mivel annál nincs fényesebb, aki felé mozogjon.

Maximumkeresési problémáknál a szentjánosbogarak fényessége egyszerűen arányos lehet célfüggvénnyel azonban minimumkeresési problémáknál a fitnesz függvényhez hasonló függvényt kell alkalmaznunk.

A firefly algoritmust alapvetően folytonos problémák megoldására készítették, de speciális kikötések és átértelmezésekkel diszkretizálható, így permutációs problémák is megoldhatók vele. Az egyedek mozgását (bármely a x_i és x_j szentjánosbogár esetén a t+1 iterációban) az alább látható két összefüggéssel határozható meg:

$$x_i^{t+1} = x_i^t + \beta_0 \cdot \exp\left(-\gamma \cdot r_{ij}^2\right) + \alpha_t \cdot \epsilon_t \tag{19}$$

$$\beta = \beta_0 \cdot e^{-\gamma r} \quad x_i = x_i \cdot (1-\beta) + x_i \cdot \beta + \alpha(rand - \tfrac{1}{2}), \tag{20}$$

ahol β_0 maximális attraktivitási érték;

γ abszorpciós koefficiens: Általában az értéke 0.1 és 10 közé esik. Egy másik meghatározása lehet a $\gamma = \frac{1}{\sqrt{L}}$, ahol L a probléma mérete;

r_{ij} az *i*-edik és a *j*-edik szentjánosbogár távolsága (metrikákra lásd példát a távolságmérés fejezetben).

α_t randomizációs paraméter: A lépések nagyságát, jelen alkalmazás esetében a cserék számát adja meg.

ϵ_t randomizációs paraméter: Egy véletlen szám vagy vektor, amelyet legtöbbször gauss vagy egyenletes eloszlás generátorral gyártunk.

Az algoritmus alkalmazását C# programnyelven készítettük el. A program egy adatfájlból olvassa be az optimalizálási probléma bemenő paramétereit: a feltöltendő objektumok koordinátáit vagy útvonalmátrixait, a kiinduló töltöttségi szinteket és az átlag fogyást napi szinten. Meg kell adni a lépésközt, a napok számát, a szentjánosbogaraink (változatok) számát és az iterációs számot. Ha megadtunk minden kezdőértéket az *Optimize* gombra kattintva elkezdődik a megadott probléma optimális számának meghatározása, amelynek előrehaladását megfigyelhetjük a nyomógomb alatti állapotjelzőn. A *First step* opcióval beállíthatjuk, hogy hány százaléktól kezdje a vizsgálatot a program. Beírhatunk saját magunk egy értéket, de választhatjuk a maximum csökkenést is, amely a legnagyobb átlagfogyást veszi alapul. A Best solution kiírás alatt a legjobb eredmény található és hozzá a minimális töltésszint van megadva százalékban. A teljes eredménylistát a *Solution vector* tartalmazza.

A program futtatását egy tesztfeladattal ellenőriztük, amelynek az alapadatai az 1. táblázatban találhatóak meg. A tesztfeladat az általános modellből származtatható le és az alapvető funkcióknak a firefly algoritmussal való összekapcsolását vizsgáljuk meg benne. A tesztfeladatban jelenleg csak egy központi raktárból indulhat ki járat és egyszerre csak egy járműre tudja meghatározni az optimális útvonalat, figyelmen kívül hagyva a jármű teherbírását. A továbbiakban tervezzük, hogy a programot kiterjesszük az általános modell minden részének vizsgálatára.

```
Begin
Induló töltöttség (IND), átlag fogyás (ATL), koordináták (X,Y,[Z]) vagy
útmátrix (L) megadása vagy kiszámítása
Lépésköz (S), iteráció (I), napok számának (D) és szentjánosbogár popula-
ció számának (X) megadása
Célfüggvény meghatározása: f(x)
for i=1:100/S
Aktuális minimum meghatározása: AKTS= 1+i*S
     for j=1:D
     Útvonal elemeinek meghatározása:
         if (IND_j<AKTS),
          Az elem az útvonalban van
         end if
      Random útvonalváltozatok generálása: SOL[]
       for k=1:elemszám+1
       Útvonalhosszak megadása: M[]=L(SOL[k];SOL[k+1])
       end for k
       Fényesség függvényének meghatározása: M_ij, M_ij∝f(X), vagy egyszerűen
       M_ij=f(X)
       for l=1:I (iterációs szám)
              Legfényesebb (minimum) változat kiválasztása
              A többi változat megvizsgálása hány elemben tér el a sor-
              rendje a legjobbtól: A(X)
              Random szám generálása: B=Rand(1:A(X)-1)
              Minden a legjobbtól különböző változatban B-szer cserélni
              egymás mellett lévő random elemeket
              A legjobb változatban egyszer cserélni egy random elemet
              Fényesség újbóli meghatározása: M_ijl
       end for l
      napi legjobb útvonal kiválasztása: M_ij
  end for j
  Adott töltöttségre vonatkozó összes nap útvonalainak összege: M_i=SUM
[M_ij]
end for i
Legkevesebb útvonallal járó töltöttség kiválasztása
eredmények feldolgozása, vizualizáció;
end
```

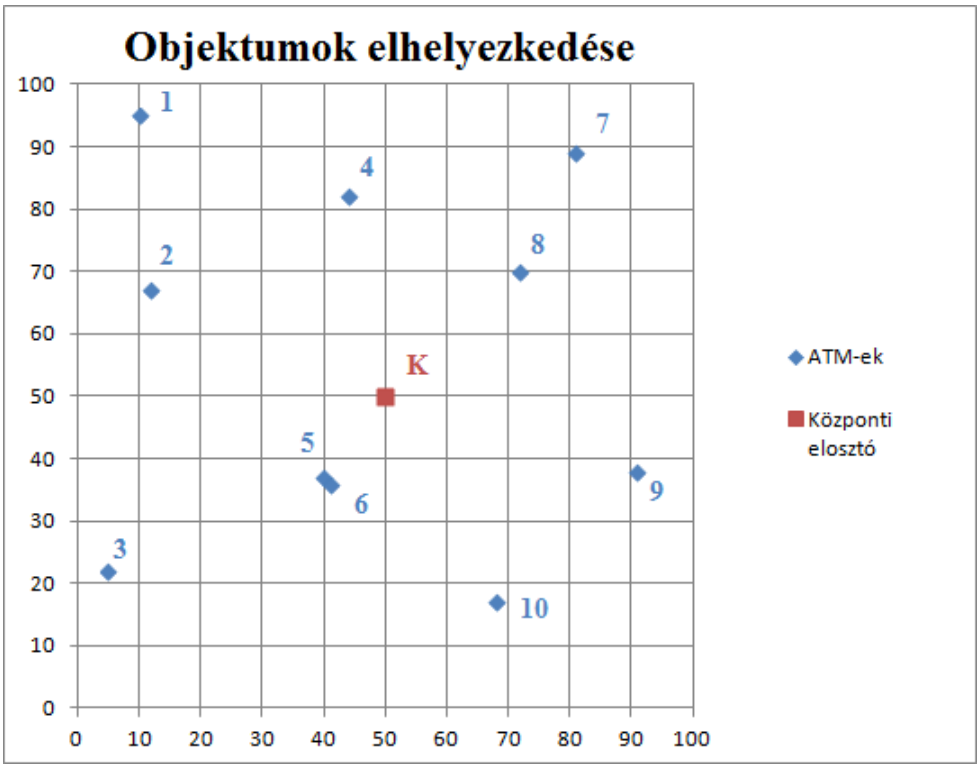
2.ábra A firefly alapú optimalizálási algoritmus pszeudokódja

Az optimalizálási feladat az általános modellből levezethető olyan feladat, ahol a cél szétszórtan elhelyezkedő objektumok feltöltési folyamatának optimalizálása. Ebben az esetben az objektumok ATM terminálok, az elhelyezkedésüket pedig a 3. ábra mutatja. A kékkel jelölt helyek a feltöltendő egységeket jelentik és piros négyzettel van jelölve a központi elosztó helység, ahonnan minden nap kimegy és visszatér egy jármű. Az objektumok közötti utat, az egyszerűség kedvéért, légvonalban számoljuk. A táblázatban megtalálható és az adatforrásban is kötelezően meg kell adni az objektumok induló töltöttségét és a napi átlagos csökkenést. Az induló töltöttséget, ha nagy időszakra vizsgálunk, nem kell pontosan megadni, ugyanis olyankor csak minimálisan befolyásolja az eredményt.

Ezek mellett még kötelező megadni a következő adatokat: az objektumok koordinátái (ilyenkor a program légvonalban összeköt minden pontpárost és meghatározza az azok közötti úthosszakat); az útmátrixot, amelyet ha egy térképadatbázisból kérünk le pontosabb adatokkal tud szolgálni, mint a kiszámítás.

Ezek után be kell állítani a program futását befolyásoló paramétereket. A lépésközt kettőre választottuk. Ilyenkor a program két százalékonként emeli a kritikus töltési szintet és minden értéknél lefuttatja az adott időtartamnak megfelelően. A napok számához egy évet, azaz 365 napot

definiáltunk, ez már nagy időtartamnak számít. Tíz darabos szentjánosbogár populációval és 100-as iterációs számmal dolgozott az algoritmus, amely egy ilyen kisméretű feladatnál maximálisan kiszolgálja az igényeket.

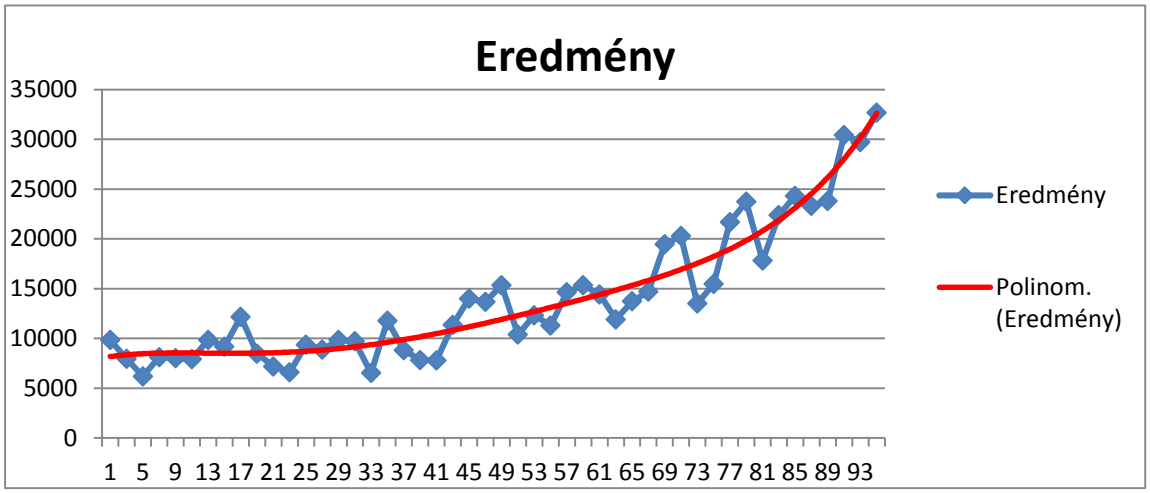


3. ábra Objektumok elhelyezkedése

Objektumok	X koordináta	Y koordináta	Induló töltöttség	Napi átlagos csökkenés
K	50	50	100	0
1	10	95	47	9
2	12	67	12	25
3	5	22	83	11
4	44	82	75	7
5	40	37	40	23
6	41	36	61	18
7	81	89	32	22
8	72	70	47	16
9	91	38	25	14
10	68	17	56	8

1. táblázat Alapadatok

A kapott számítási eredmények: A legkisebb szállítási munkát 7%-os riasztási szintnél lehet elérni, de 23% és 35%-nál is minimumot vesz fel a diagram. Emellett a pontokra nagyon jól lehet illeszteni egy polinom görbét, amely azt mutatja, hogy 30~35%-ig alacsony szállítási munkával lehet kiszolgálni az objektumokat, majd utána exponenciálisan megugrik a szállítási távolság.



4. ábra Szállítási útvonalak hossza a feltöltési szintek függvényében

Mint ahogy azt a futtatási eredmények is mutatják, az algoritmus jól alkalmazható a definiált modellben megfogalmazott problémák megoldására. Természetesen az itt bemutatott modell egy alapmodellnek tekinthető, a további kutatási irányok még számos kihívást tartogatnak a valóságos körülményeket minél inkább figyelembe vevő modellek megalkotásával [15].

ÖSSZEFOGLALÁS

A szolgáltatási folyamatok tervezése speciális modelleket és azok megoldására alkalmas módszereket igényel. Jelen kutatómunka keretében a szerzők egy olyan firefly alapú heurisztikus optimalizálási algoritmust dolgoztak ki, melynek segítségével különböző hálózatszerűen működő szolgáltatási tevékenység logisztikai folyamata optimalizálható.

KÖSZÖNETNYILVÁNÍTÁS

A kutatómunka a Miskolci Egyetem stratégiai kutatási területén működő Logisztikai, Informatikai, Mechatronikai Kiválósági Központ keretében valósult meg.

FELHASZNÁLT IRODALOM

[1] SHARAD N. KUMBHARANA, GOPAL M. PANDEY: Solving travelling salesman problem using firefly algorithm. International Journal for Research in Science & Advanced Technologies. Issue 2. Volume 2. 2013. pp. 53-57

[2] M. SEVAUX, K. SÖRENSEN: Permutation distance measures for memetic algorithms with population management. Proceedings of the sixth metaheuristics International Conference, Austria. 2005. pp. 1-8

[3] S. ZHANG, C. K. M. LEE, H. K. CHAN, K. L. CHOY, Z. WU: Swarm intelligence applied in green logistics: A literature review. Engineering Applications of Artificial Intelligence. Volume 37, January 2015, pp. 154–169

[4] R. KÁSA, Á. GUBÁN: Business Process Amelioration Methods, Techniques and their Service Orientation: A Review of Literature. In: Vastag Gy (ed.) Research in the Decision Sciences for Global Business. Upper Saddle River: Pearson, 2015. pp. 219-238.

[5] Á. GUBÁN Á. Z. MEZEI, Á. SÁNDOR: Service Processes as Logistic Workflows. In: Branko Katalinic (ed.) DAAAM International Scientific Book 2014. Bécs: DAAAM International. 2014. pp. 485-500.

[6] W. LIU, Y. WANG: Quality control game model in logistics service supply chain based on different combinations of risk attitude. International Journal of Production Economics, Volume 161. 2015. pp. 181-191.
[7] Y.H. VENUS LUN, K.-H. LAI, C.W.Y. WONG, T. C .E. CHENG: Greening propensity and performance implications for logistics service providers. Transportation Research Part E: Logistics and Transportation Review. Volume 74. 2015. pp. 50-62.
[8] A. DE MARCO, A. C. CAGLIANO, G. MANGANO, F. PERFETTI: Factor Influencing Logistics Service Providers Efficiency' in Urban Distribution Systems. Transportation Research Procedia. Volume 3. 2014. pp. 499-507.
[9] A. HOFF, H. ANDERSSON, M. CHRISTIANSEN, G. HASLE, A. LØKKETANGEN: Industrial aspects and literature survey: Fleet composition and routing. Computers & Operations Research. Volume 37. 2010. pp. 2041-2061.
[10] P. EVANGELISTA: Environmental sustainability practices in the transport and logistics service industry: An exploratory case study investigation. Research in Transportation Business & Management. Volume 12. 2014. pp. 63-72.
[11] C.-N. LIAO, H.-P. KAO: An evaluation approach to logistics service using fuzzy theory, quality function development and goal programming. Computers & Industrial Engineering, Volume 68. 2014. pp. 54-64
[12] R. O. LARGE, N. KRAMER, R. K. HARTMANN: Procurement of logistics services and sustainable development in Europe: Fields of activity and empirical results. Journal of Purchasing and Supply Management. Volume 19. Issue 3. 2013. pp. 122-133.
[13] W. LI, Y. ZHONG, X. WANG, Y. CAO: Resource virtualization and service selection in cloud logistics. Journal of Network and Computer Applications. Volume 36. Issue 6. 2013. pp. 1696-1704.
[14] F. GZARA, E. NEMATOLLAHI, A. DASCI: Linear location-inventory models for service parts logistics network design. Computers & Industrial Engineering. Volume 69. 2014. pp. 53-63.
[15] BÁNYAI Á: Optimisation of intermediate storage network of JIT purchasing. Advanced Logistic Systems. Volume 5. 2011. pp.35-40.